Auteurs : Teodoro Mounier Tebas & Diane Thierry

Master 2 : Économétrie et Statistiques

Enseignant : Tanguy Le Nouvel

Année : 2020-2021

Méthodes de scoring et management du risque

Détection de fraude

Analyse des transactions financières par téléphone en Afrique, avec Python

UNIVERSITÉ DE NANTES

IAE NANTES
ÉCONOMIE & MANAGEMENT

Sommaire

- Introduction
- Manipulations préliminaires de la base
- Statistiques descriptives
- Modélisations
- Conclusion

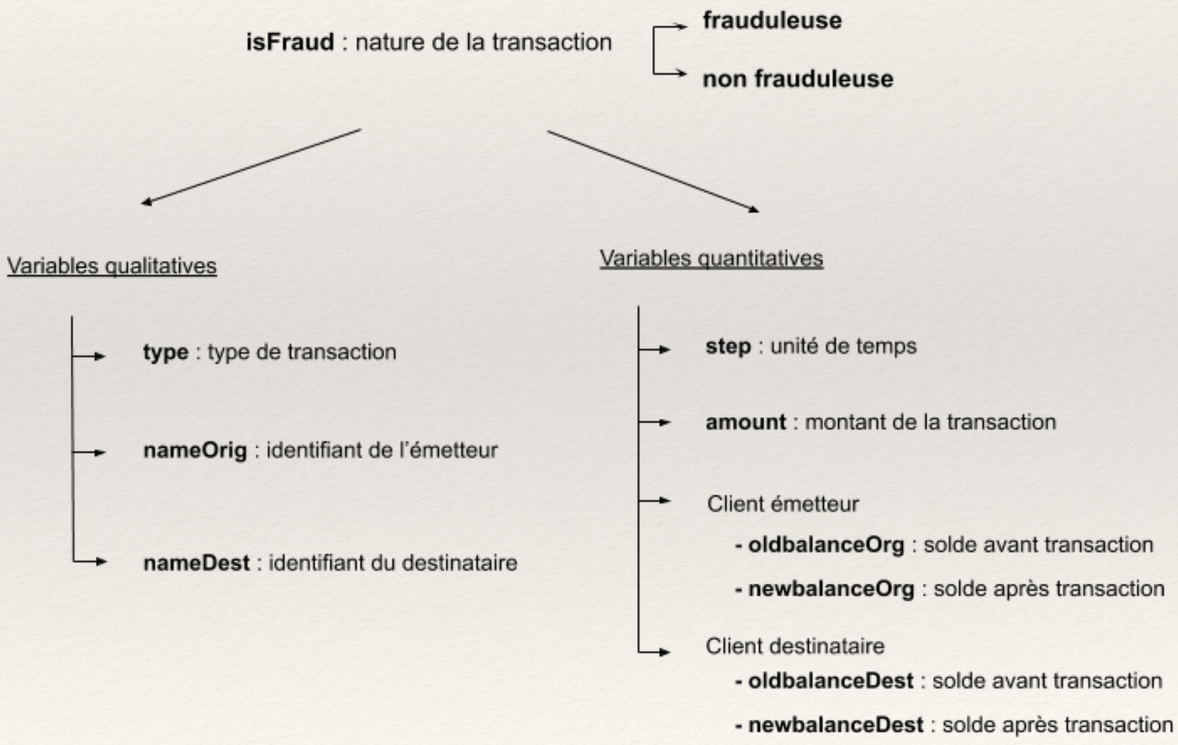
Introduction

- La **bancarisation de l'Afrique** par les téléphones mobiles est un phénomène croissant depuis la fin des années 2000 ; en 2020 le continent regroupe un total de 500 millions d'utilisateurs de mobile money. Cependant, la mise en place de systèmes de paiement par téléphone induit un nombre de fraudes important et des pertes conséquentes pour les organismes. La détection et la réduction des fraudes est donc devenu un objectif majeur pour les organismes qui sont contraints de rembourser les victimes de fraude, ce qui représente des dizaines de millions d'euros par an.

- L'analyse réalisée vise ainsi à **améliorer le système de détection de fraude** grâce à l'utilisation de modèles de machine learning en temps réel sur 750.000 transactions pour lesquelles nous disposons de plusieurs informations.

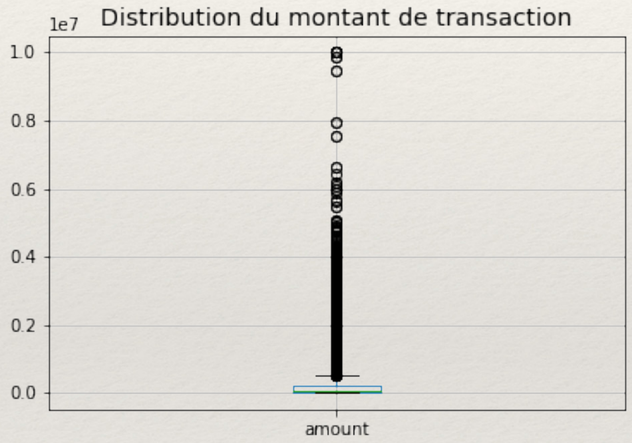
Les données

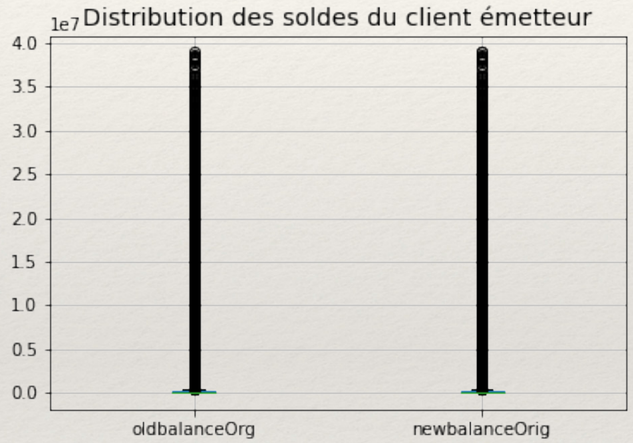
- Le **jeu train** contient 500.000 observations
- Le **jeu test** contient 250.000 observations

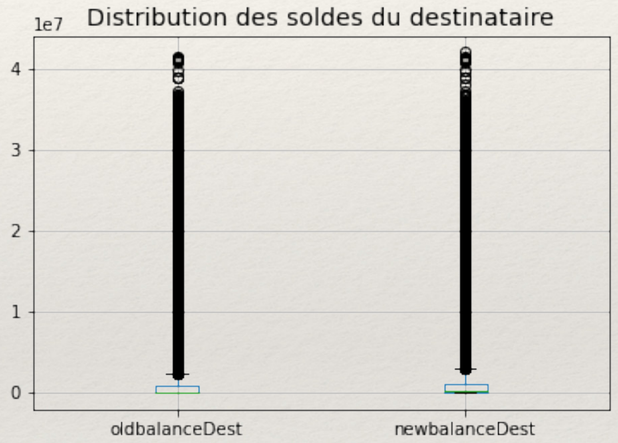


Manipulations préliminaires de la base

Avant toute analyse exploratoire des données il est important de les rendre propres pour avoir une étude valide par la suite, cela passe par la détection et la correction des valeurs manquantes et des points atypiques.



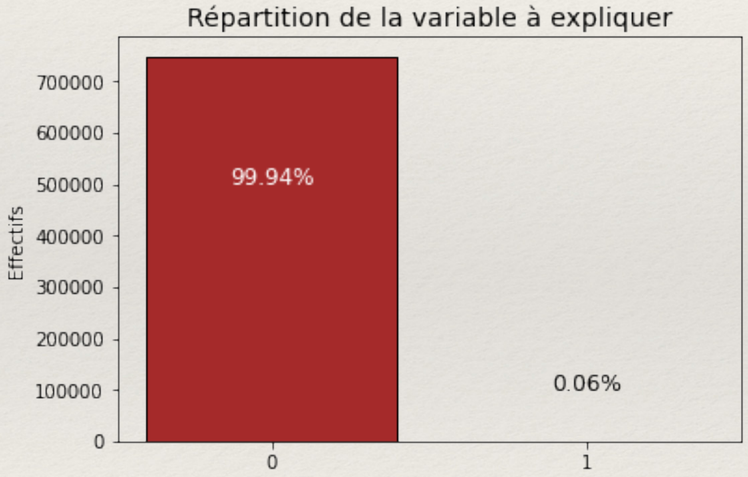




Il n'y a aucune **valeur manquante** pour les 10 variables de la base. Pour les **points atypiques**, à partir des boxplots on voit que certains montants de transactions sont très élevés, en ordonnant la base on voit qu'il s'agit de 7 transactions qui sont toutes frauduleuses et donc que nous décidons de garder dans la base pour en exploiter les informations. On suppose dès à présent que plus le montant de la transaction est élevé et plus le risque de fraude est grand. Pour les soldes des clients émetteurs et destinataires de la transaction on ne voit aucun point qui se détache des autres, on les laisse telles quelles.

Statistiques descriptives

Nous devons rééquilibrer la base pour que les algorithmes d'apprentissage puissent identifier correctement les déterminants de la modalité cible (Y=1), qui dans notre cas est la détection de fraude. L'écueil à éviter serait de penser que l'on obtient de bonnes estimations avec 99% de bien prédits, alors que le modèle ne fait que prédire des non-évènements (Y=0) présents à 99% dans la base. L'objectif initial qui vise à détecter les fraudes ne serait donc pas atteint.

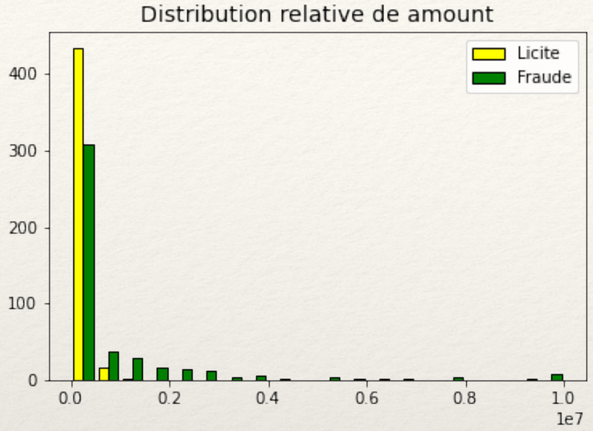


Construction d'un échantillon équiréparti entre Y=1 et Y=0

- à partir des bases **train** et **test** rassemblées
- on garde toutes les transactions frauduleuses
- on en prend aléatoirement le même nombre pour Y=0
- on rassemble le tout en un data frame pour l'analyse

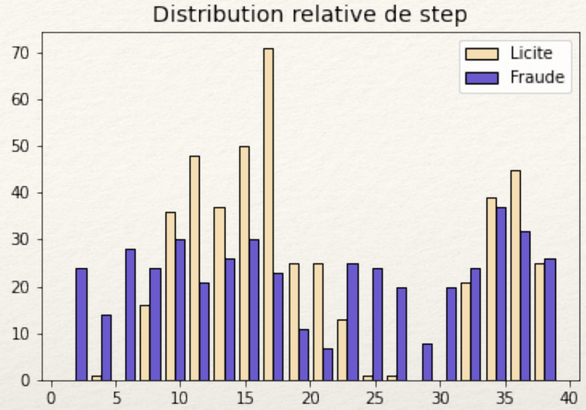
Transactions → licites : 749.546 / frauduleuses : 454 → *transformation* → licites : 454 / frauduleuses : 454 → On obtient une base avec 908 observations

Représentations graphiques des variables selon qu'il y ait fraude ou non



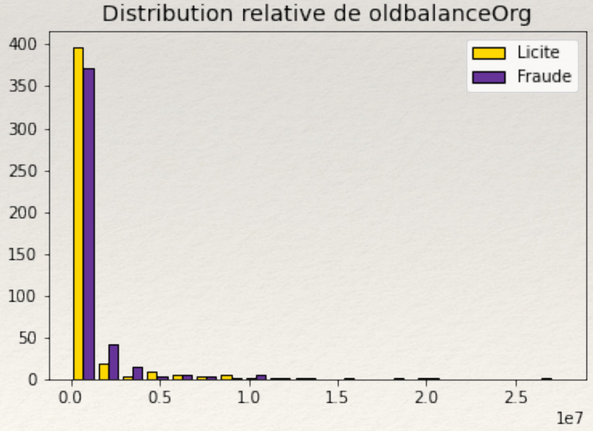
- **Amount**

Proportionnellement, il y a plus de fraudes quand le montant de la transaction est élevé.



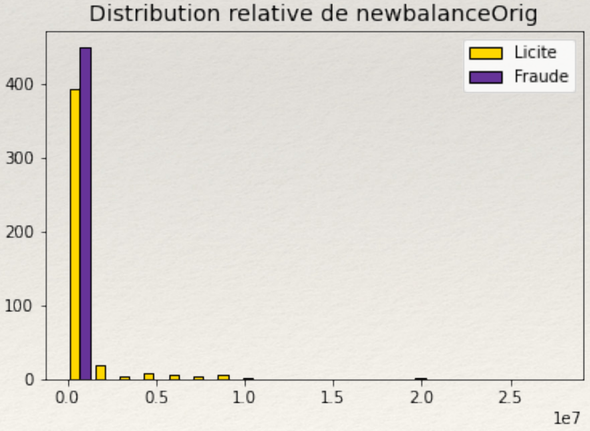
- **Step**

*Proportionnellement au nombre de transactions licites qui sont surtout en journée, il y a plus de fraudes la nuit.**



- **OldbalanceOrg**

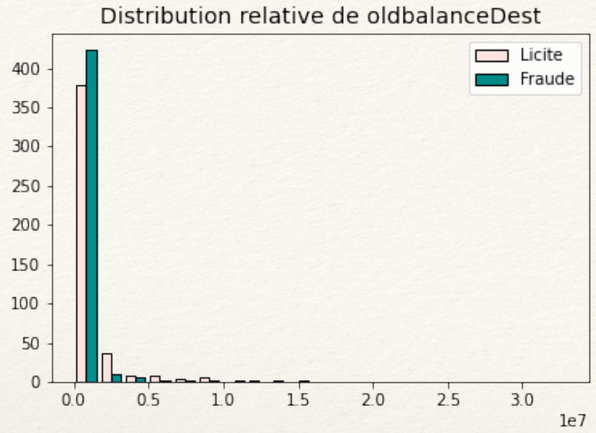
Même constat que pour le montant de la transaction, il faudra trouver le seuil auquel taux de fraude > taux licite.



- **NewbalanceOrig**

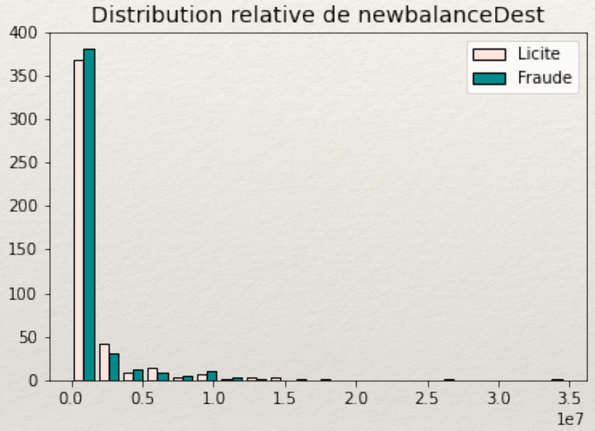
Cette fois le nombre de fraudes est plus grand lorsque le solde du client émetteur de la transaction est faible.

** Nous avons considéré que les valeurs correspondaient aux heures la journée telles que 0=minuit, sur deux jours.*



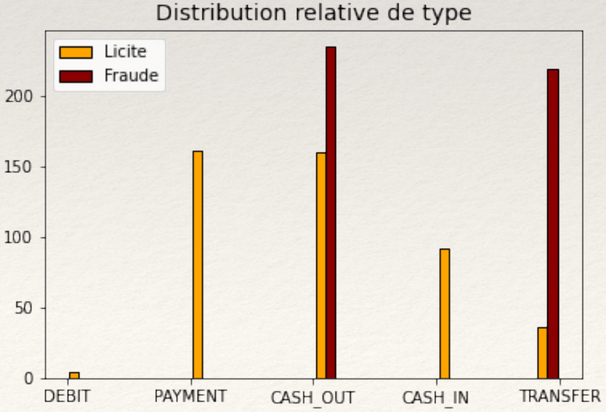
- **OldbalanceDest**

Plus de fraudes quand le solde du destinataire avant la transaction financière est faible.



- **NewbalanceDest**

Cette fois on n'identifie pas vraiment de pattern : la différence des transactions frauduleuses ou licites selon le nouveau solde du destinataire ne semble pas significative.



- **Type**

Les types de transactions avec le plus de fraudes sont les transferts et les retraits d'argent.

- **Conclusion sur ces différentes statistiques**

On voit que pour chaque variable (excepté '*nameOrig*') il y a un seuil à partir duquel les fraudes excèdent les transactions licites, ou une modalité dans laquelle les fraudes sont généralement plus nombreuses. Cela nous amène à créer de nouvelles variables binaires qui prennent la valeur 1 lorsque la fraude est la plus importante, on a :

- **is_bigamount** : à partir de *amount*
- **is_night** : à partir de *step*
- **is_bigoldsoldorig** : à partir de *oldbalanceOrg*
- **is_lownewsoldorig** : à partir de *newbalanceOrig*
- **is_lowoldsolddest** : à partir de *oldbalanceDest*
- **typeDest** : à partir de *nameDest*

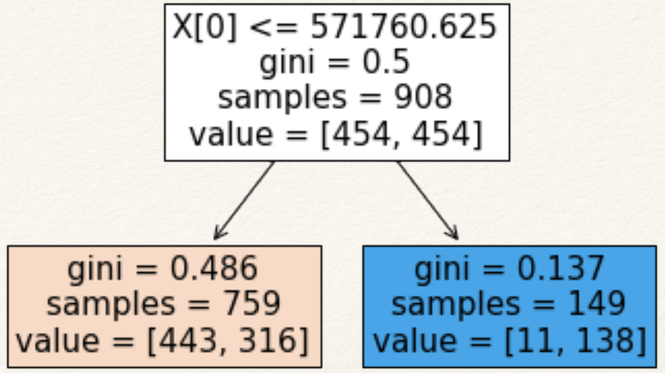
On ne construit pas de nouvelle variable à partir du **type** de la transaction puisque s'agissant déjà d'une qualitative avec des modalités, construire une binaire n'apporterait pas d'information. À partir de '*nameDest*' nous avons récupéré la lettre (**C** pour client et **M** pour commerçant) et nous avons créé **typeDest** qui indique si la personne est client ou commerçant.

- **Développement de la démarche pour le montant de la transaction**

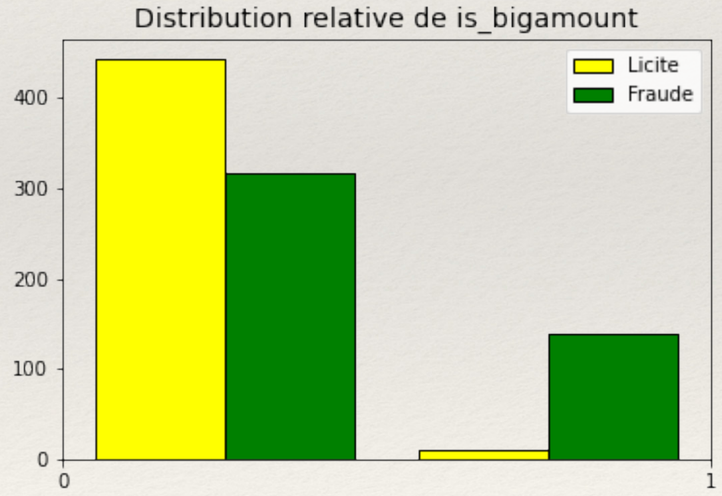
Les histogrammes nous ont permis de voir qu'à un certain seuil le nombre de fraudes devenait supérieur au nombre de transactions non frauduleuses, nous avons donc décidé de créer une variable qui accentue cet effet à partir de '*amount*'. Nous avons choisi de construire un arbre de décision pour obtenir ce seuil : l'arbre coupe le jeu de données à partir de la variable avec une homogénéité intra-groupe et une grande hétérogénéité inter-groupe. Autrement dit, les caractères entre les groupes seront opposés et en ajoutant Y à expliquer, les 2 groupes correspondront aux groupes optimisés avec le plus et le moins de fraude relativement au montant de la transaction.

On crée donc une variable '**is_bigamount**' qui prend la valeur 1 si le montant est supérieur à 571.760,63 et 0 sinon. Grâce à l'histogramme de distribution de cette nouvelle variable binaire on retrouve un pourcentage de fraude nettement supérieur pour des montants élevés (X=1), ce qui confirme sa pertinence.

On fait de même pour toutes les autres variables pour lesquelles on a observé des patterns, on les code de manière à avoir toujours le nombre plus important de fraudes en X=1.



Arbre de décision



Distribution de la variable transformée

Sélection de variables au vu des modélisations

Après avoir construit 6 nouvelles variables à partir des constats émis avec les statistiques descriptives, on prépare la base aux futures modélisations :

- on transforme les 7 variables qualitatives en indicatrices (**dummies**)
- on **standardise** les 5 variables quantitatives grâce à la commande *preprocessing.StandardScaler()*

Seulement après ces modifications nécessaires à l'application des modèles, nous procédons à la sélection des variables que nous inclurons dans ces derniers ; nous voulons éviter tout problème de corrélation qui biaiserait les coefficients estimés.

La **matrice de corrélation** nous montre une corrélation forte (coefficient > 0,8) entre les soldes des clients émetteurs et destinataires avant et après transaction (*balanceDest* et *balanceOrig*). Les autres variables n'ont pas de corrélations trop fortes que nous devrons corriger avant les estimations. Concernant les variables binaires créées à partir des variables quantitatives visibles sur la matrice, nous supposons que les **dépendances** sont de même nature et devrons donc choisir une des 2 variables à inclure entre *is_lownewsoldorig* et *is_bigoldsoldorig*. À cause des corrélations et/ou de la redondance de nos variables suite à la création de certaines, nous procédons à une sélection à partir des méthodes **Lasso** et **Elastic-Net**.

	LASSO	ELASTIC-NET	Sélection finale
Temps *step*	**0**	**0**	X
Montant *amount*	0.128	0.801	✓
Montant>571761 *is_bigamount*	0.013	0.097	✓
Solde émetteur avant *oldbalanceOrg*	**0**	**0**	X
Solde>102 *is_bigoldsoldorig*	**0**	1.07	✓
Solde émetteur après *newbalanceOrig*	-0.118	-0.511	✓
Solde<24 *is_lownewsoldorig*	**0**	-0.938	✓
Solde destinataire avant *oldbalanceDest*	0.266	1.508	X
Solde<162992 *is_lowoldsolddest*	-0.024	**0**	X
Solde destinataire après *newbalanceDest*	0.015	1.869	✓
Type de transaction * *type*	0.163	0.803	✓
Type de destinataire *typeDest*	0.134	1.069	✓
Transaction faite de nuit *is_night*	**0**	-0.284	✓

** La valeur correspond aux retraits d'argent qui sont les plus présents dans l'échantillon*

: variables initiales

: variables créées

- La régression régularisée **LASSO** rend nuls les coefficients des variables jugées non significatives pour les modèles, on voit que 5 ont été tronquées à 0 dont 3 que nous avons crées.
- La méthode **EN** compile les avantages des Ridge et Lasso en proposant un coefficient relatif a l'impact des variables sur Y, donc en ne prenant pas au hasard une variable lorsqu'il y a corrélation forte avec une autre.
- Finalement, au vu des sélections de variables ainsi que des corrélations constatées avec la matrice, nous décidons d'écarter les variables '*step*', '*oldbalanceOrg*', '*oldbalanceDest*' et '*is_lowoldsolddest*'.

Pour les estimations nous aurons donc 9 facteurs explicatifs de la fraude, et 908 transactions.

Modélisations

Pour chacun des modèles que nous construirons à partir des données d'entraînement, nous évaluerons sa qualité grâce au score qui donne le **taux de bonnes prédictions** ; nous le regarderons à 3 niveaux :

- taux sur échantillon d'entraînement : lorsque le modèle est appliqué sur les données sur lesquelles il s'est construit
- taux en validation croisée : les données sont divisées en k sous-groupes de tailles égales, le modèle s'entraîne sur k-1 groupes et s'évalue sur le dernier, cette méthode pallie au problème de sur-apprentissage en ré-echantillonant le jeu
- taux sur échantillon test : le modèle est appliqué aux données inconnues du jeu test

Le modèle final que nous retiendrons pour l'évaluation des scores (probabilité de fraude) sera celui qui maximise ces taux de bien classés. Pour construire et évaluer sur des données différentes nous partitionnons les données en 2 jeux par tirage aléatoire stratifié ; le jeu **train** comprend alors 635 observations et le jeu **test** en comprend 273.

Pour l'ensemble des modèles que nous appliquons nous trouvons la fonction dans la librairie *sklearn*. Dans un premier temps nous construisons une **régression logistique** faisant partie des modèles économétriques qui s'appliquent lorsque l'on analyse une variable qualitative binaire, puis nous estimons 2 modèles de machine learning pour lesquels nous cherchons au préalable les hyper paramètres avec *GridSearch*, dans le but d'optimiser les modélisations. Il s'agit des méthodes de **forêt aléatoire** et de **réseau de neurones** (Perceptron multi-couches). Nous sauvegarderons les prédictions de chacun d'entre eux dans un dataframe.

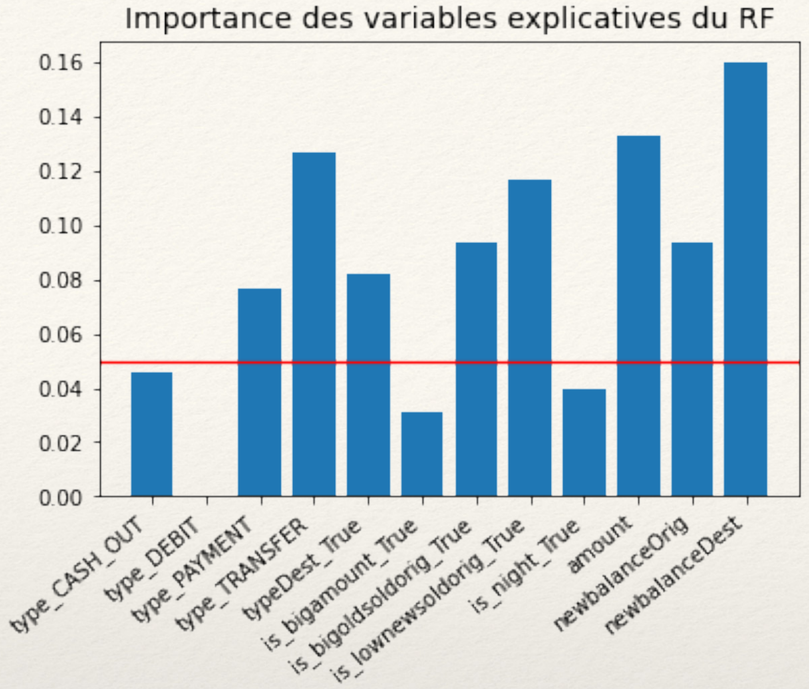
- De manière générale on voit qu'il n'y a pas de **sur apprentissage** (lorsque le modèle se calque trop aux données d'entraînement) sur les modèles de ML, dans le sens où les taux de prédictions sur les échantillons train et test diffèrent peu.
- On voit aussi que la régression logistique qui est un modèle 'simple' par rapport au machine learning, offre des résultats satisfaisants puisque sur données inconnues elle prédit correctement la nature de la transaction (frauduleuse ou licite) plus de 89 fois sur 100.

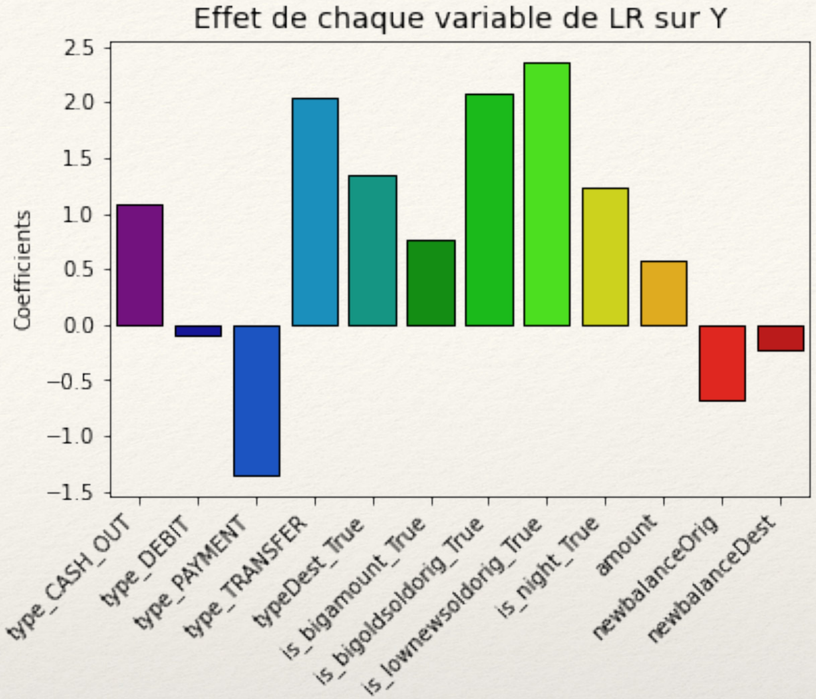
- Taux de bonnes prédictions associés aux modèles -

	Régression logistique	**Random forest**	**Multi couches Perceptron**
Hyper-paramètres	/	- *max_depth=20* - *min_samples_split=0.03* - *n_estimators=200*	- *batch_size=100* - *max_iter=200* - *tol = 0.0001*
• Train	0.861	0.882	0.85
• CV (k=5)	0.865	0.877	0.852
• Test	0.894	0.905	0.890

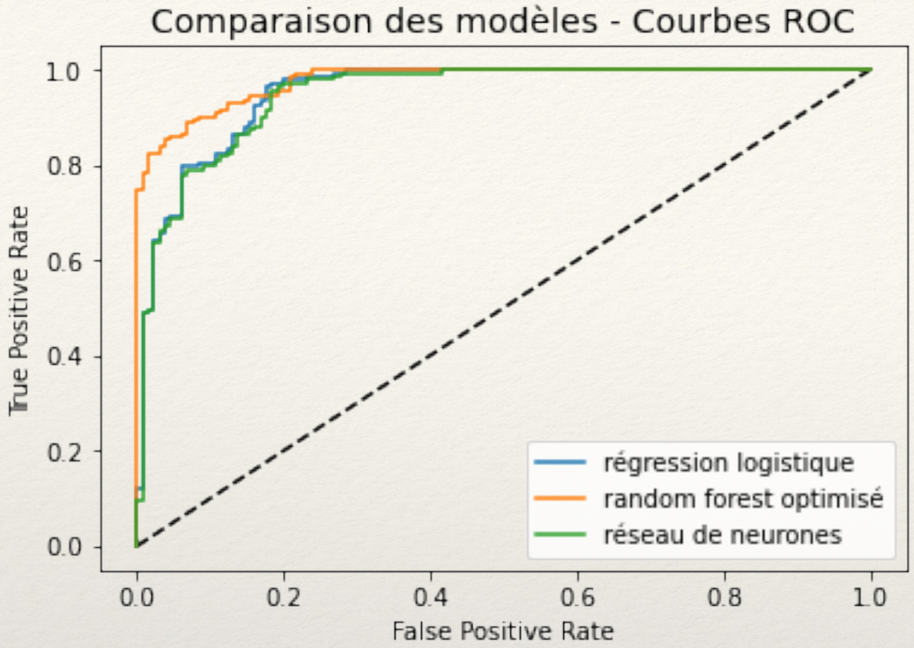
- Le meilleur modèle, c'est-à-dire celui qui minimise les erreurs de prédiction, est le **Random forest optimisé** par recherche en grille qui prédit correctement 9 transactions sur 10. On voit cependant que l'écart de bien prédits sur le jeu test est très faible entre nos 3 modèles.
- Pour le RF, l'**écart des taux** de bonnes prédictions hors et sur données d'entraînement est de seulement 0.023%. Les hyper-paramètres qui contrôlent la complexité des modèles, s'ils sont bien paramétrés, permettent d'en optimiser les performances. Ceux qui optimisent la forêt aléatoire sont : une profondeur d'arbre de 20, une fraction du nombre minimum de découpes de 0.03 avec 200 arbres estimés.

Les facteurs explicatifs de la fraude





- Grâce aux différentes modélisations nous pouvons définir quels facteurs sont importants dans l'explication de la nature des fraudes. À partir des modèles de ML, c'est la commande '*feature_importances_*' sous python qui nous offre un aperçu de cela : nous regardons celles du Random Forest qui constitue notre meilleur modèle. On voit alors que seules 4 variables ne sont pas significativement importantes dans l'explication de la fraude (seuil de risque de 5% signifié par la ligne rouge) ; le fait que la transaction soit un retrait d'argent (pourtant modalité où le nombre de fraudes était grand), que ce soit un débit, que le montant soit supérieur à 571.761 et que le transfert ait été fait de nuit. Toutes les autres variables sont importantes pour Y.
- La **régression logistique** offre la possibilité de connaître l'effet de chaque variable sur le phénomène à expliquer (par le calcul d'effets marginaux et odd ratios), nous voyons alors que celle qui a le plus grand impact est une variable que nous avons créée à savoir *is_lownewsoldorig* donc un client dont le solde après transaction est inférieur à 24, a 2.36 plus de chance de frauder qu'un client dont le solde est plus élevé. De même, un client qui fait un transfert a 2.04 fois plus de chance de frauder, par rapport à un client faisant un dépôt d'argent (modalité CASH-IN de référence). On voit enfin que le fait de faire un paiement réduit le risque de fraude ce qui se comprend du fait de la sécurité autour des paiements en magasins.



Sur le graphique qui reprend les **courbes ROC** associées à chaque modélisation, on fait le même constat à savoir que le meilleur modèle est celui de la forêt aléatoire puisqu'on voit que ses performances sont les mieux : c'est le modèle pour lequel il y a le moins de mal classés (vrais ou faux positifs).

	Y*=1	**Y*=0**
Y=1	125	5
Y=0	21	122

- Matrice de confusion du RF -

On peut justement regarder la matrice de confusion associée à ce modèle (random forest) ; on voit que sur 273 transactions le modèle en a classé correctement 247. En revanche, il a considéré 21 transactions comme frauduleuses alors qu'elles ne l'étaient pas (**faux positifs**), et n'a pas réussi à détecter 5 qui l'étaient bel et bien (**vrais positifs**). On voit alors qu'une simple prédiction binaire peut être limitée, surtout dans des cas concrets comme celui-là où ici 21 clients se seraient vu interdire la transaction alors qu'elle était licite. Survient alors la méthode de **scoring** qui vise à affecter un score à un client ; il s'agit de la probabilité que l'évènement se réalise ; que la transaction soit frauduleuse dans notre cas. À partir de cette probabilité, l'objectif est de déterminer le plus justement possible un seuil à partir duquel une transaction doit être considérée comme frauduleuse et donc rejetée par l'organisme financier.

Conclusion

Depuis l'import de notre base nous avons manipulé les données pour les rendre exploitables pour les estimations, nous les avons observées avec les statistiques descriptives à partir desquelles nous avons créé de nouvelles variables, puis nous avons modélisé les fraudes par une régression logistique, une forêt aléatoire et un réseau de neurones. Le meilleur modèle c'est-à-dire celui qui minimise les erreurs de prédiction, est le RF avec un taux de bien prédit sur jeu test de 90.5%. Nous avons ensuite pu observer les effets de chaque variable explicative sur Y, puis à partir de chaque modèle nous avons récupéré les scores ainsi que les prédictions, que nous avons stockés avec Y observé dans une base finale qui reprend les résultats de notre analyse ; disponible **ici**.

On y voit les scores et les prédictions de chaque modèle, le nom du client effectuant la transaction et sa nature réelle (frauduleuse ou non frauduleuse). À partir de cela nous avons établi la règle suivante, pour une détection efficace des fraudes :

- **score < 0.27** : jusqu'au score de 27% la transaction n'est pas frauduleuse
- **0.27 < score < 0.73** : dans cet intervalle la fraude est probable mais demande une vérification « humaine »
- **score > 0.73** : à partir de 73% la transaction est frauduleuse et mieux vaut l'interdire pour éviter tout problème

On dit alors que jusqu'au score de 57 on considère la transaction non frauduleuse puis à partir de 57 on la considère comme une fraude. Nous avons établi ces seuils en comparant les scores prédits par le modèle RF ainsi que les prédictions binaires, à la nature observée de la transaction (Y). Le code couleur de ces intervalles mis sur le fichier permet de constater les différents faux ou vrais positifs selon chaque modèle estimé. Ainsi cette analyse aura permis de détecter les fraudes des transactions financières par téléphone en Afrique, qui sont nombreuses et tendent à croître davantage encore.